Para Exergames

Alexandre Melo
André Bonetto

24/09/2025
17:00 às 21:00h
Laboratório F101

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Agenda 13:30 às 17:30 (4 horas)

1. **Pygame**
2. Console T-TEA
3. OpenCV
4. MediaPipe
5. Pygame + OpenCV + MediaPipe

Objetivo do Curso

Pygame

Pygame é um conjunto de módulos Python projetados para construir jogos de videogames.

- Portátil e roda em quase todas as plataformas e sistemas operacionais;

- Pygame é gratuito. Lançado sob a licença LGPL, você pode criar jogos de código aberto, freeware, shareware e comerciais com ele;

- CPUs multi-core podem ser usadas facilmente;

História (Mais de duas décadas)

Pygame foi escrito originalmente por Pete Shinners pelos idos do ano 2000…

Python Gaming with Pygame

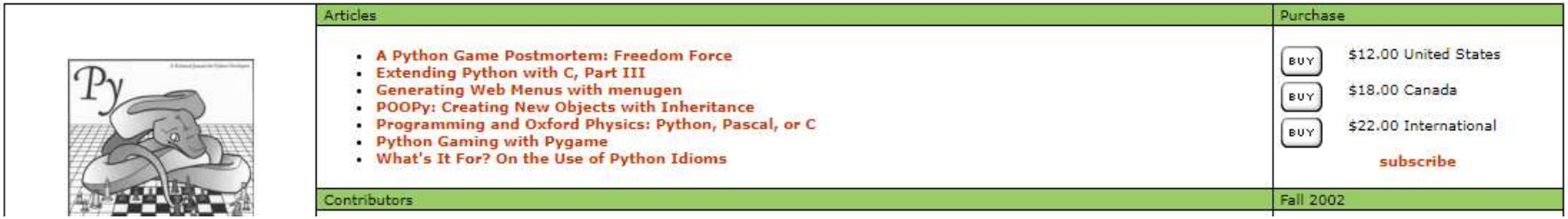
Pete Shinners

Abstract:

After a year and half of development, I released Pygame 1.5 earlier this year. You may have heard of Pygame before, or you may have playedvone of its games. As its name suggests, Pygame is a collection of Python modules and extensions designed for use in game development. An experienced C programmer, I discovered Python and SDL at about the same time during the summer of 2000. At that time the current version of Python was 1.5.2. SDL, the Simple Directmedia Library, created by Sam Lantinga, is a cross-platform C library for controlling multimedia, similar to DirectX. SDL has been used for hundreds of commercial and open source games. I was impressed at how clean and straightforward Python and SDL were, and it didn't take long before I realized mixing Python and SDL was an interesting idea.

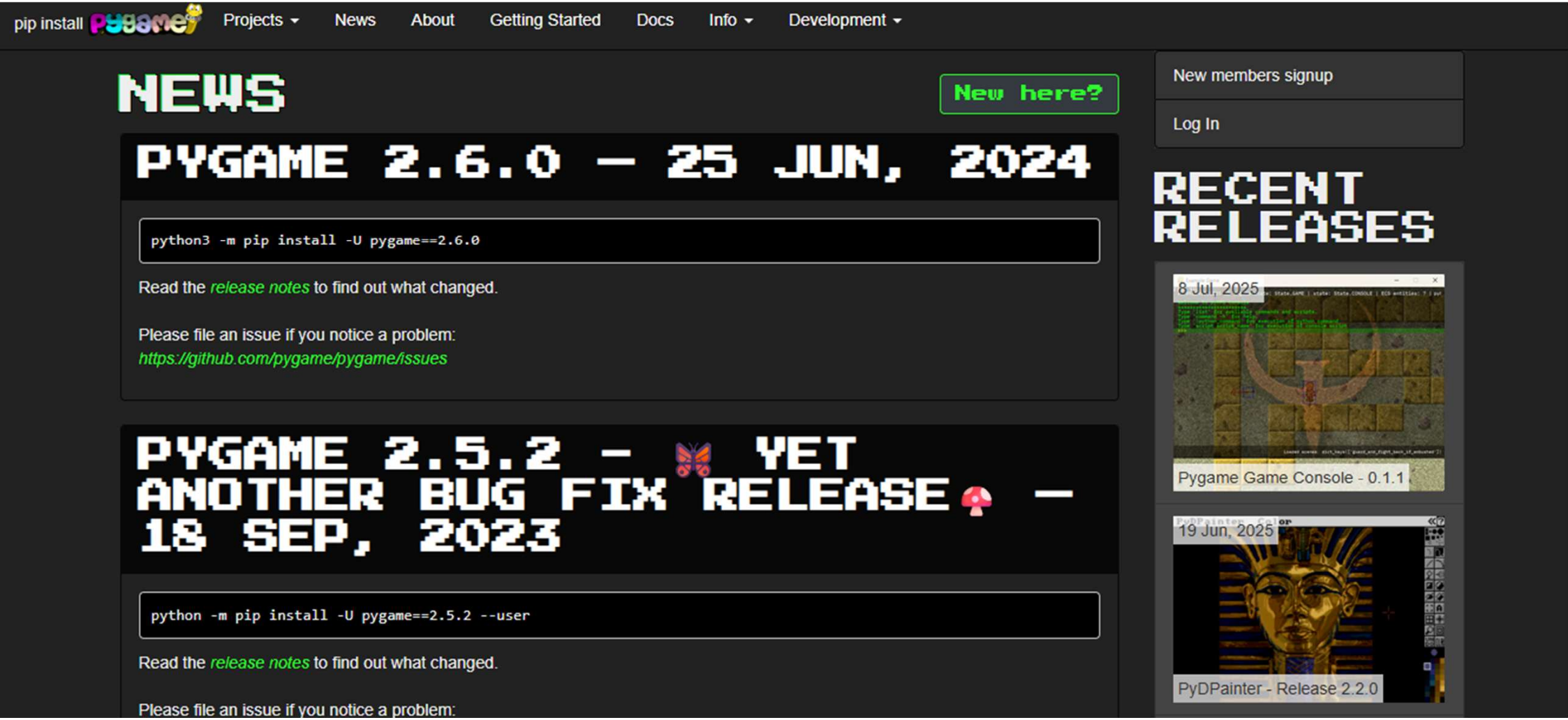
I soon discovered a small project with exactly the same idea, PySDL, already under development by Mark Baker. PySDL was a straightforward implementation of SDL as a Python extension. The interface was cleaner than a generic SWIG wrapping, but I thought it forced a procedural style. The sudden death of PySDL prompted me to start a new project combining Python and SDL, with the goal that simple things were easy to do, and difficult things weren't too hard either. I started Pygame in October, 2000, releasing version 1.0 six months later.

From the Issue:



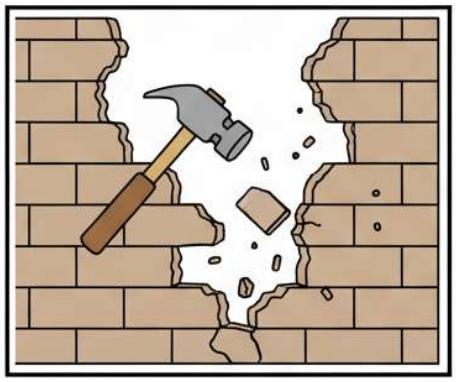
https://web.archive.org/web/20030810011958/http://store.pyzine.com:80/article.phtml?a=2

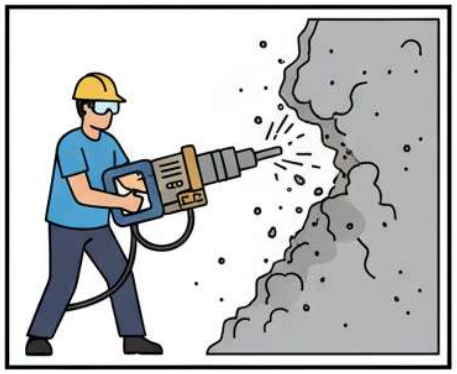
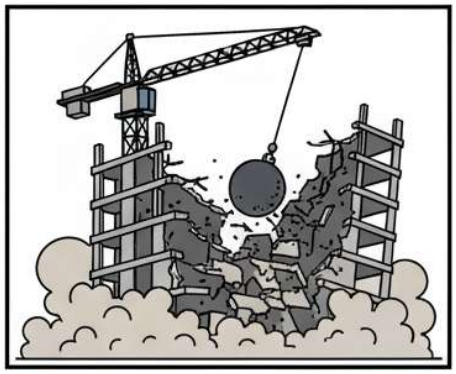
www.pygame.org



Pygame & Gaming

"Pygame é adequado para jogos?"

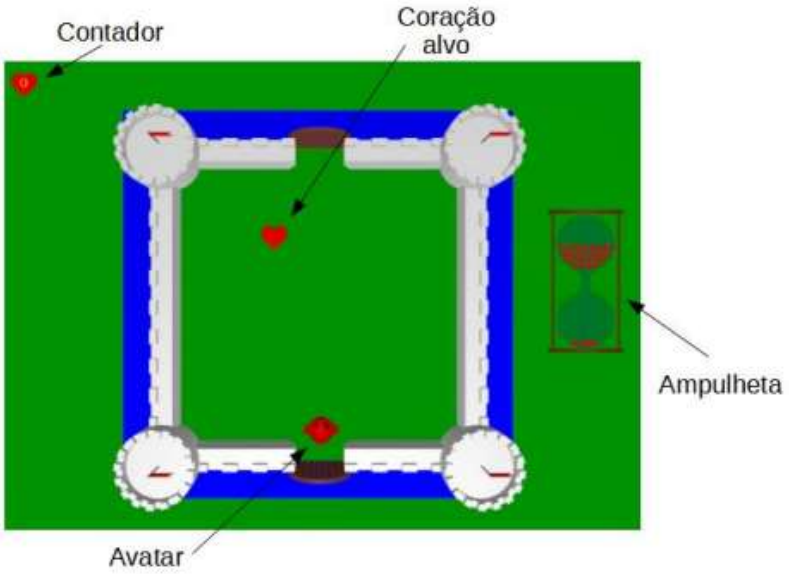


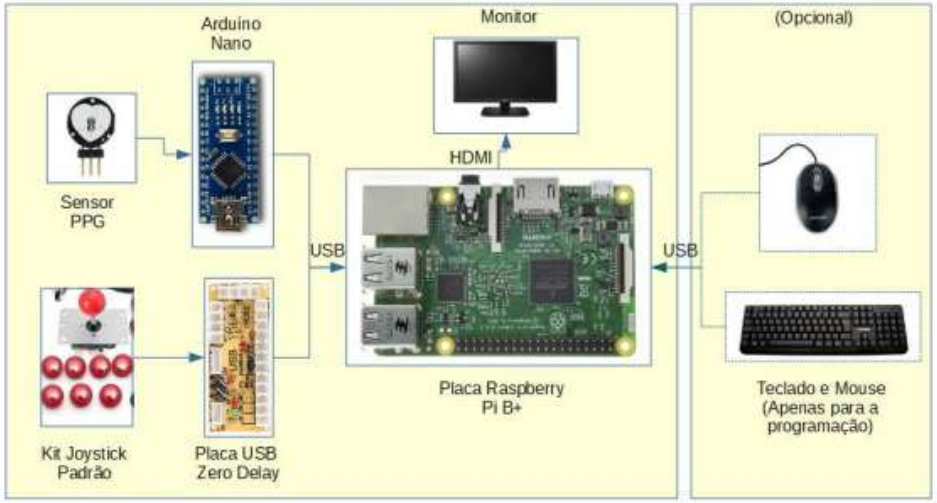



A resposta é: “Depende do jogo e seus objetivos”.

Dr. Core


Spectrum Games
Dr. Core
60 cm
60 cm
40 cm


Contador
Coração alvo
Ampulheta
Avatar

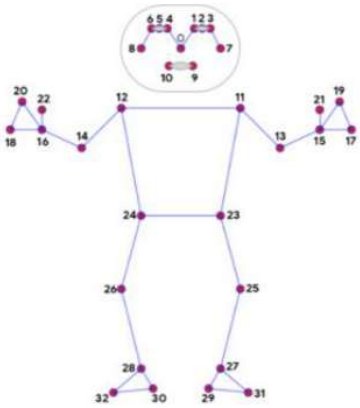

Arduino Nano
Monitor
(Opcional)
Sensor PPG
HDMI
USB
USB
Kit Joystick Padrão
Placa USB Zero Delay
Placa Raspberry Pi B+
Teclado e Mouse (Apenas para a programação)

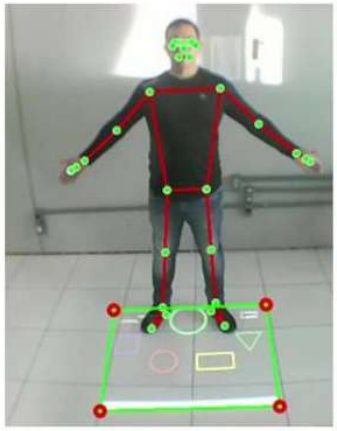

BPM: 83
Usuário A


BPM: 66
Usuário B

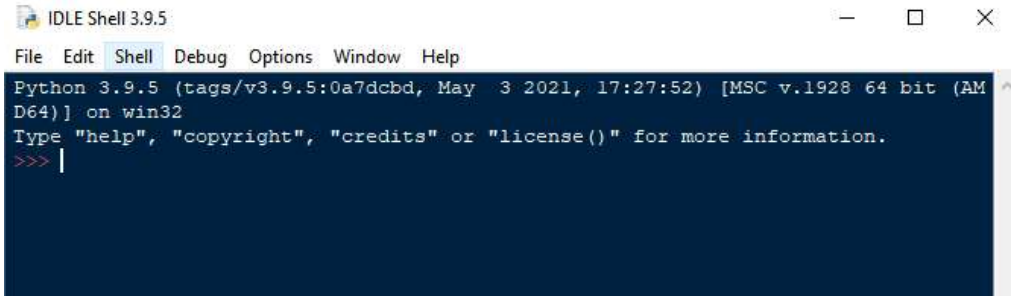

pygame

Jogos para o console T-TEA





Interfaces de Desenvolvimento



Instalação Pygame

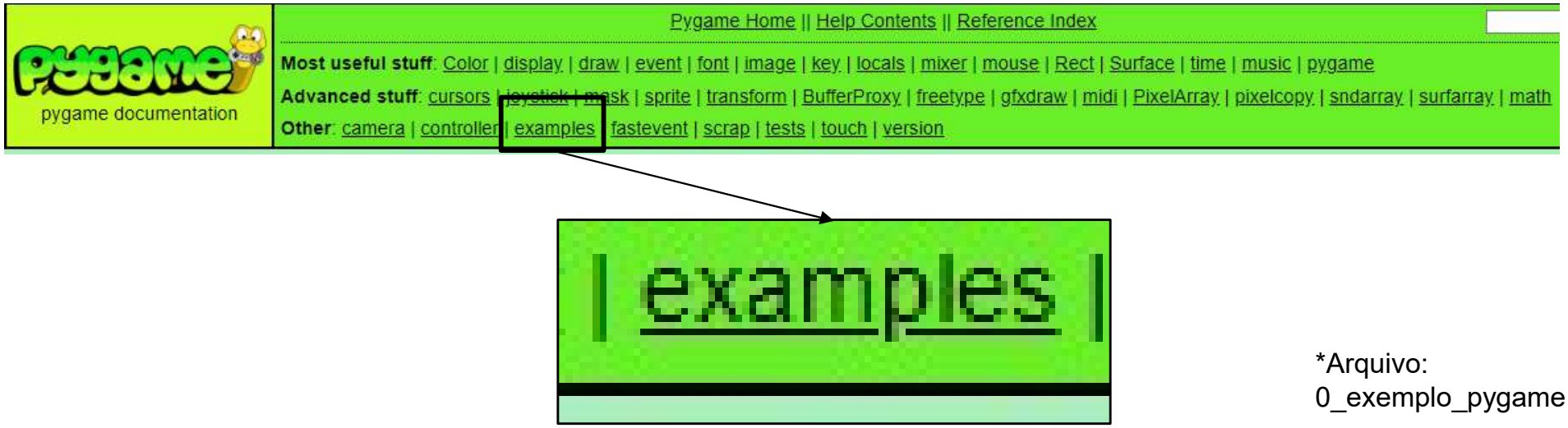
INSTALAR

Abra um prompt de comando e digite: ***pip show pygame***

Pygame & Gaming*

Digite na interface de desenvolvimento:

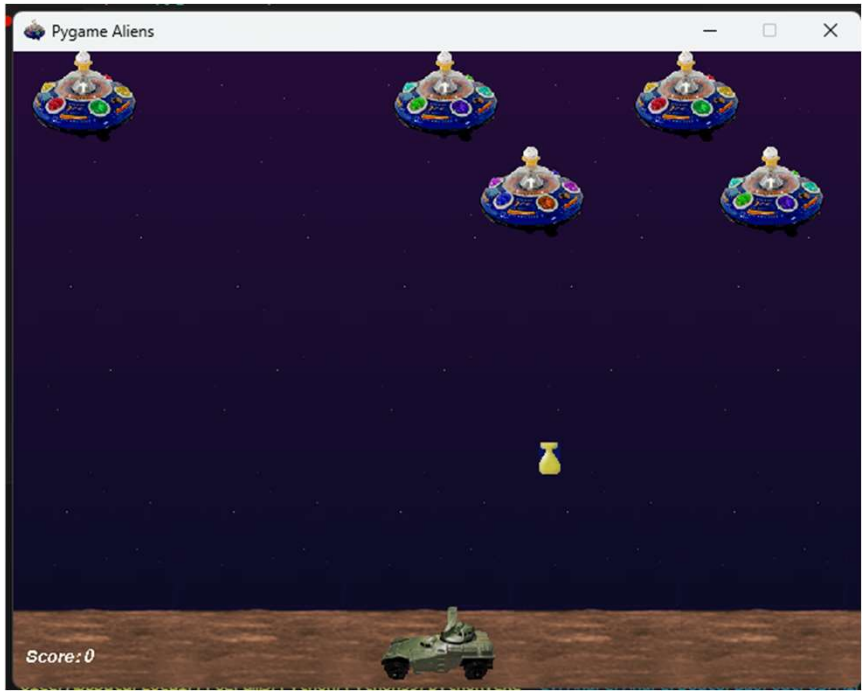
```
import pygame.examples.aliens
pygame.examples.aliens.main()
```



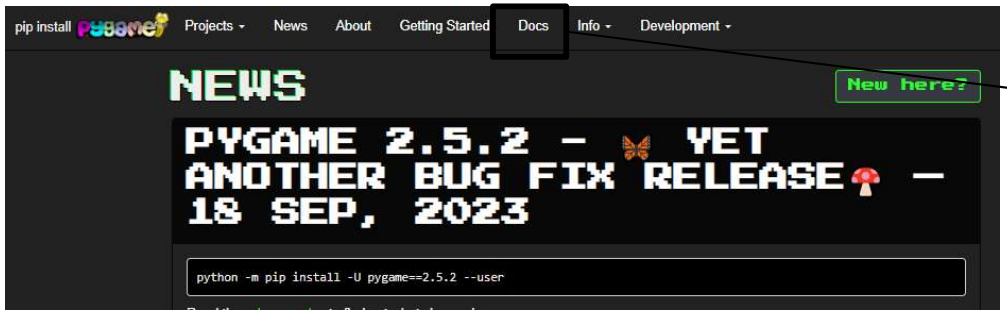
*Arquivo:
0_exemplo_pygame.py

Pygame & Gaming*

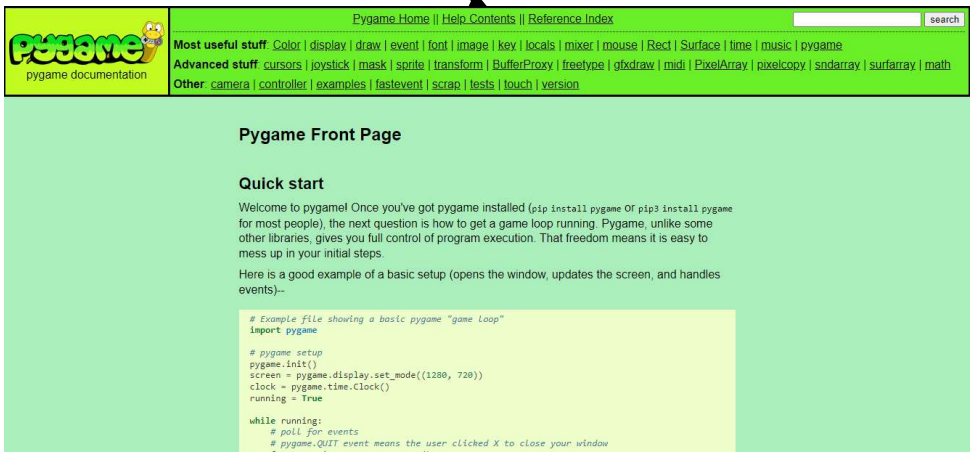
```
import pygame.examples.aliens
pygame.examples.aliens.main()
```



Vamos Experimentar



https://www.pygame.org/news



https://www.pygame.org/docs/

1 - Loop Básico

```python
# Example file showing a basic pygame "game loop"
import pygame

# pygame setup
pygame.init()
screen = pygame.display.set_mode((1280, 720))
clock = pygame.time.Clock()
running = True

while running:
    # poll for events
    # pygame.QUIT event means the user clicked X to close your window
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # fill the screen with a color to wipe away anything from last frame
    screen.fill("purple")

    # RENDER YOUR GAME HERE

    # flip() the display to put your work on screen
    pygame.display.flip()

    clock.tick(60)  # limits FPS to 60

pygame.quit()
```

EXERCÍCIO:

a) Troque o fundo para azul;

Exemplo disponível em:
https://www.pygame.org/docs/

1 - Loop Básico

```python
# Example file showing a basic pygame "game loop"
import pygame

# pygame setup
pygame.init()
screen = pygame.display.set_mode((1280, 720))
clock = pygame.time.Clock()
running = True

while running:
    # poll for events
    # pygame.QUIT event means the user clicked X to close your window
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # fill the screen with a color to wipe away anything from last frame
    screen.fill("purple")

    # RENDER YOUR GAME HERE

    # flip() the display to put your work on screen
    pygame.display.flip()

    clock.tick(60)  # limits FPS to 60

pygame.quit()
```

Black
Blue
White
Yellow
Green
Red
Brown

2 – Mover a bolinha

```python
# Example file showing a circle moving on screen
import pygame

# pygame setup
pygame.init()
screen = pygame.display.set_mode((1280, 720))
clock = pygame.time.Clock()
running = True
dt = 0

player_pos = pygame.Vector2(screen.get_width() / 2, screen.get_height() / 2)

while running:
    # poll for events
    # pygame.QUIT event means the user clicked X to close your window
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # fill the screen with a color to wipe away anything from last frame
    screen.fill("purple")

    pygame.draw.circle(screen, "red", player_pos, 40)

    keys = pygame.key.get_pressed()
    if keys[pygame.K_w]:
        player_pos.y -= 300 * dt
    if keys[pygame.K_s]:
        player_pos.y += 300 * dt
    if keys[pygame.K_a]:
        player_pos.x -= 300 * dt
    if keys[pygame.K_d]:
        player_pos.x += 300 * dt

    # flip() the display to put your work on screen
    pygame.display.flip()

    # limits FPS to 60
    # dt is delta time in seconds since last frame, used for framerate-
    # independent physics.
    dt = clock.tick(60) / 1000
```

EXERCÍCIOS:

a) Troque a cor da bola para amarelo;
b) Troque o fundo para vermelho;
c) Mude a velocidade de movimentação da bola para 10x mais rápido e 10x mais lento;
d) Mude os controles de direção para as setas.

Exemplo disponível em:
https://www.pygame.org/docs/

2 – Mover a bolinha

```python
# Example file showing a circle moving on screen
import pygame

# pygame setup
pygame.init()
screen = pygame.display.set_mode((1280, 720))
clock = pygame.time.Clock()
running = True
dt = 0

player_pos = pygame.Vector2(screen.get_width() / 2, screen.get_height() / 2)

while running:
    # poll for events
    # pygame.QUIT event means the user clicked X to close your window
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # fill the screen with a color to wipe away anything from last frame
    screen.fill("purple")

    pygame.draw.circle(screen, "red", player_pos, 40)

    keys = pygame.key.get_pressed()
    if keys[pygame.K_w]:
        player_pos.y -= 300 * dt
    if keys[pygame.K_s]:
        player_pos.y += 300 * dt
    if keys[pygame.K_a]:
        player_pos.x -= 300 * dt
    if keys[pygame.K_d]:
        player_pos.x += 300 * dt

    # flip() the display to put your work on screen
    pygame.display.flip()

    # limits FPS to 60
    # dt is delta time in seconds since last frame, used for framerate-
    # independent physics.
    dt = clock.tick(60) / 1000
```

Black, Blue, White, Yellow, Green, Red, Brown...

Up, Down, Left e Right

Tempo...

Módulos Básicos

Pygame Home || Help Contents || Reference Index

Most useful stuff: Color | display | draw | event | font | image | key | locals | mixer | mouse | Rect | Surface | time | music | pygame

Advanced stuff: cursors | joystick | mask | sprite | transform | BufferProxy | freetype | gfxdraw | midi | PixelArray | pixelcopy | sndarray | surfarray | math

Other: camera | controller | examples | fastevent | scrap | tests | touch | version

Construir uma Tela e Sair*

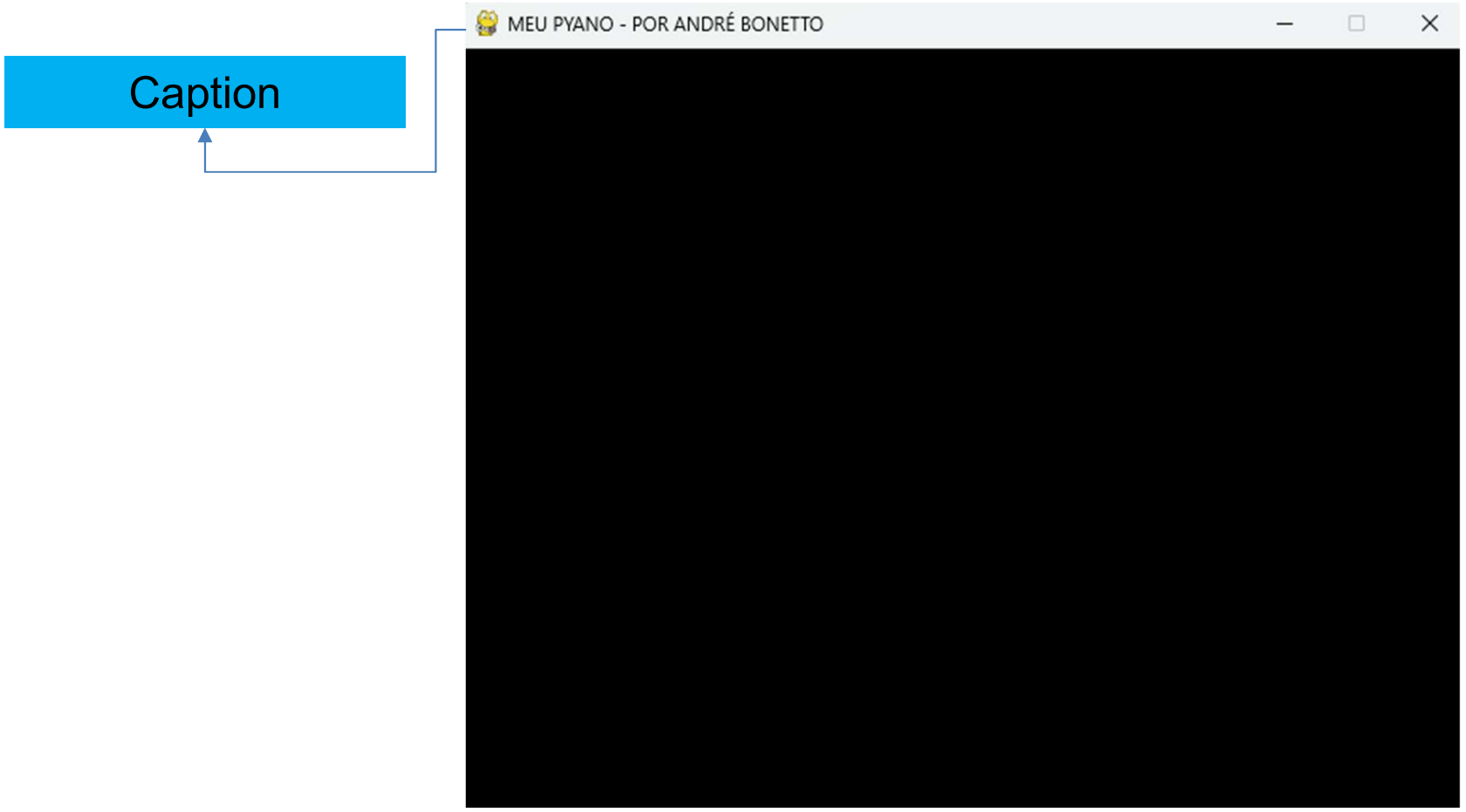
```
import pygame

pygame.init()
largura_tela = 630
altura_tela = 480
tela= pygame.display.set_mode((largura_tela,altura_tela))
pygame.display.set_caption('MEU PYANO – COLOQUE SEU NOME AQUI')
```

```
run = True

while run:


    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run=False

pygame.quit()
```

Neste laço colocamos o JOGO

*Arquivo:
1_criar_tela_e_fechar.py

Construir uma Tela e Sair*



Linha*

pygame.draw.line (tela, cor, (X e Y início), (X e Y final), espessura)

```
pygame.draw.line(tela, "white", (0,240), (640,240), 50)
pygame.display.flip()
```

O “FLIP” atualiza toda a tela de uma vez só

(X e Y início)

(X e Y fim)

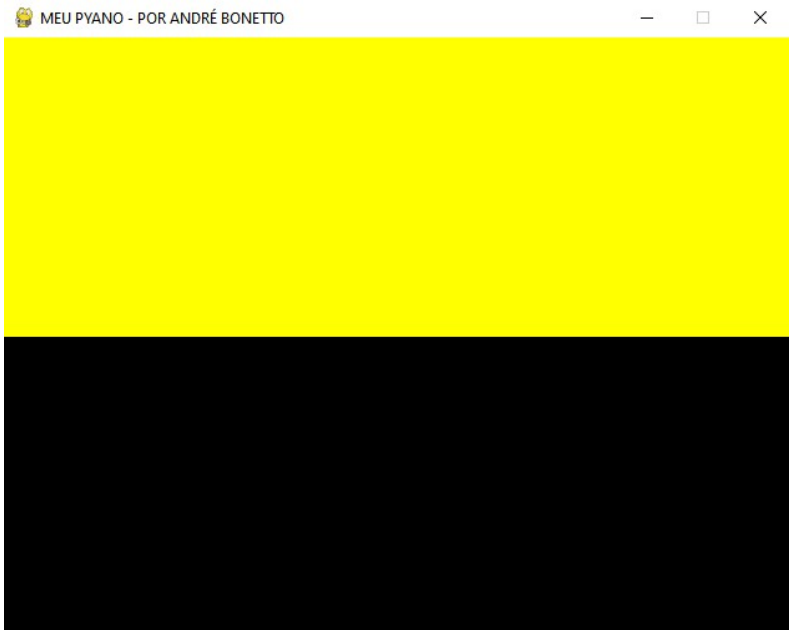
*Arquivo:
2_desenhar_linha.py

Retângulo*

pygame.draw.rect (tela, cor, (X e Y de referência), largura, altura)

```
pygame.draw.rect(tela, "yellow", [0,0,630,240])
pygame.display.flip()
```

(X e Y referência)

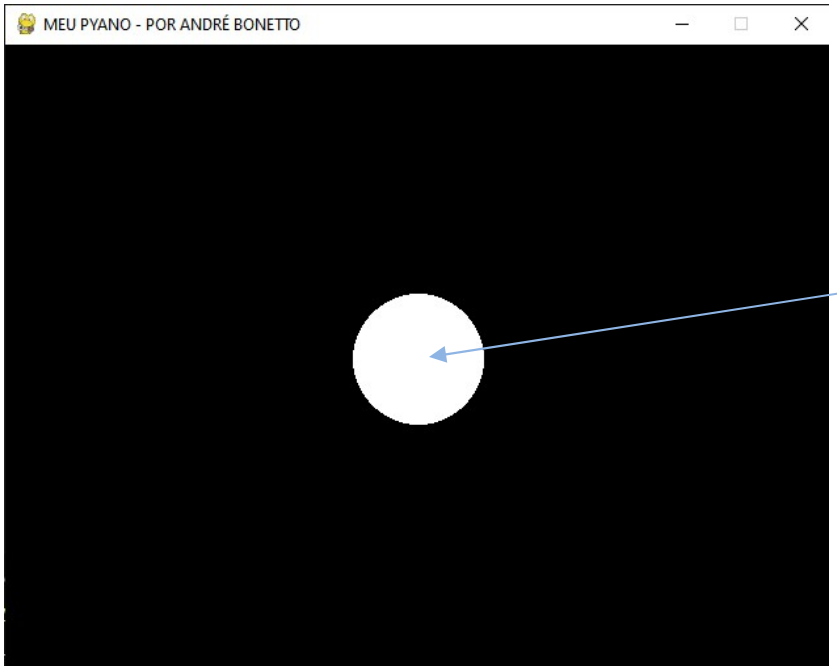


*Arquivo:
3_desenhar_retangulo.py

Círculo*

pygame.draw.circle (tela, cor, centro, raio)

```
pygame.draw.circle(tela, "white", ((largura_tela/2),(altura_tela/2)), 50)
pygame.display.flip()
```



(X e Y do centro),

*Arquivo:
4_desenhar_circulo.py

Elipse*

pygame.draw.elipse (tela, cor, retângulo que engloba a elipse)

```
pygame.draw.circle(tela, "white", ((largura_tela/2),(altura_tela/2)), 50)
pygame.display.flip()
```

*Arquivo:
5_desenhar_elipse.py

Fontes

```
fonte = pygame.font.SysFont("Arial",40, bold=True, italic=True)

texto = fonte.render('MEU PYANO', True, 'white’)

tela.blit(texto, [(180), (60)])

pygame.display.flip()
```

Pega uma fonte do sistema

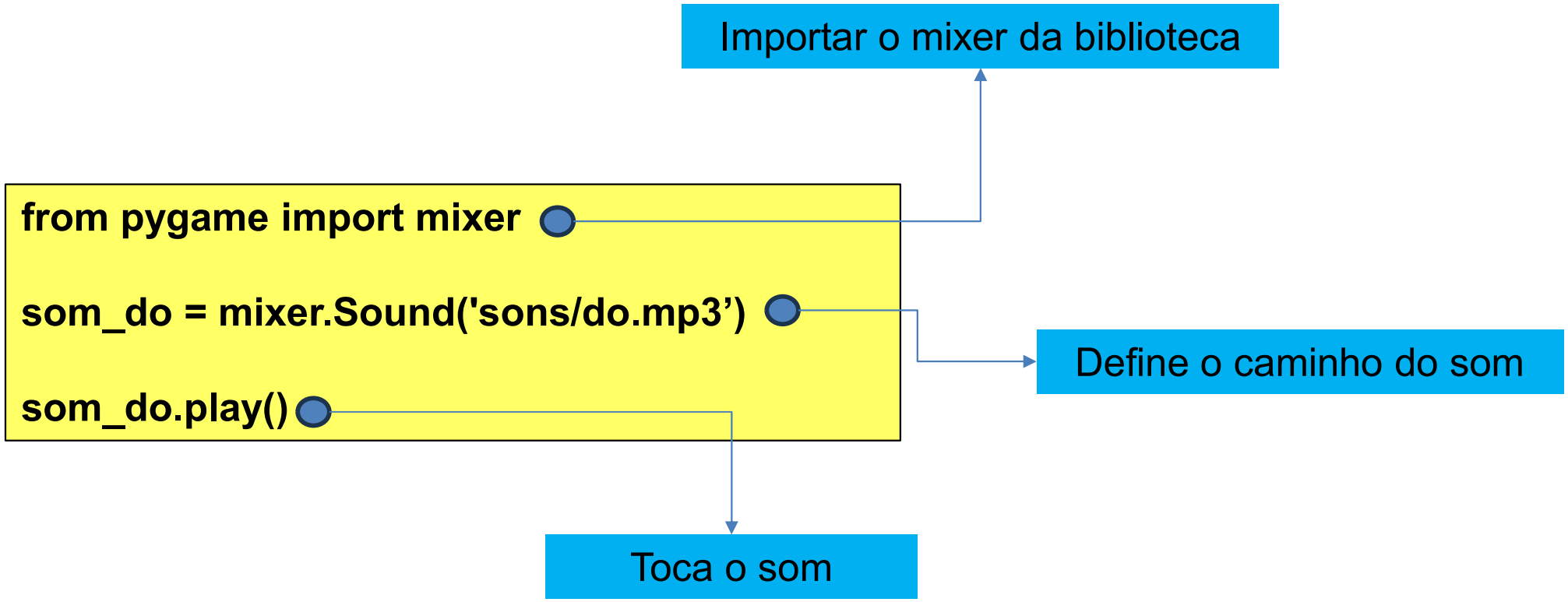
Renderiza o texto com a fonte escolhida

Adiciona o texto renderizado na superfície escolhida

Fontes*

*Arquivo:
6_desenhar_fonte.py

Sons*



*Arquivo:
7_som.py

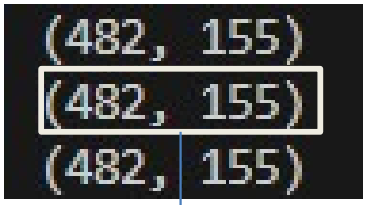
Sons*

```
import pygame
import pygame.display
from pygame import mixer

pygame.init()
largura_tela = 630
altura_tela = 480
tela= pygame.display.set_mode((largura_tela,altura_tela))
pygame.display.set_caption('MEU PYANO - POR ANDRÉ BONETTO')

som_do = mixer.Sound('sons/do.mp3')
run = True

while run:
    som_do.play()
    pygame.time.delay(1000)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run=False

pygame.quit()
```

*Arquivo: 7_som.py

Posição do Mouse*

```
import pygame

pygame.init()
largura_tela = 640
altura_tela = 480
tela= pygame.display.set_mode((largura_tela,altura_tela))
run = True

while run:

    pos = pygame.mouse.get_pos()
    print(pos)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run=False

pygame.quit()
```



A posição X,Y do mouse no terminal

*Arquivo: 8_posicao_mouse.py

Mouse (Por evento)*

```
import pygame
import pygame.display
import pygame.event

pygame.init()
largura_tela = 640
altura_tela = 480
tela= pygame.display.set_mode((largura_tela,altura_tela))
run = True

while run:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run=False
        if event.type == pygame.MOUSEBUTTONDOWN:
            print("Clicou!")
            print(event)
        if event.type == pygame.MOUSEBUTTONUP:
            print("Soltou!")
            print(event)

pygame.quit()
```

```
<Event(1026-MouseButtonUp {'pos': (409, 198), 'button': 3, 'window': None})>
Clicou!
<Event(1025-MouseButtonDown {'pos': (409, 198), 'button': 2, 'window': None})>
Soltou!
```

Em qual posição X,Y o evento ocorreu

Qual botão foi pressionado

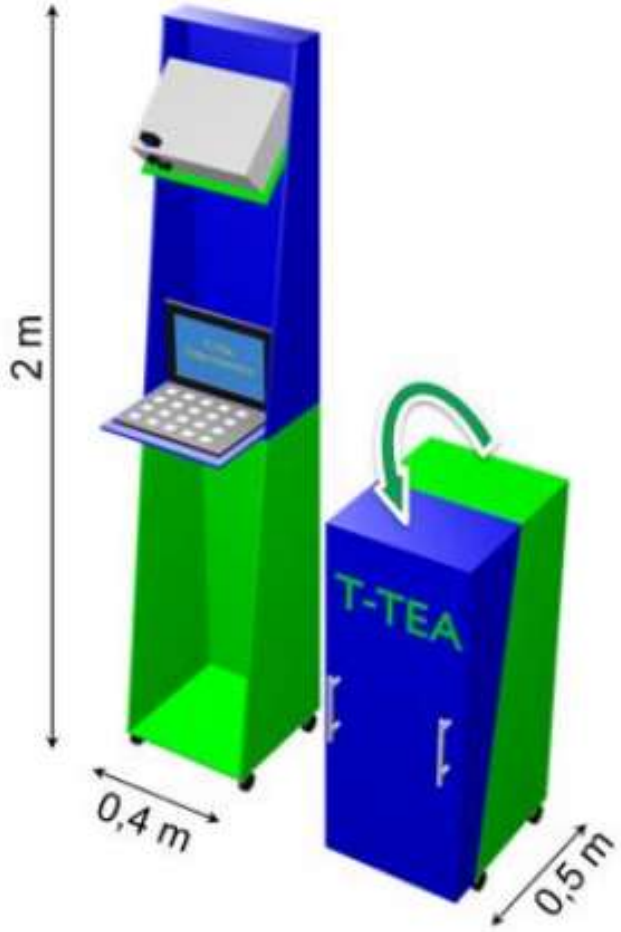
*Arquivo: 9_posicao_mouse_evento.py

Exercício: Construa a Interface

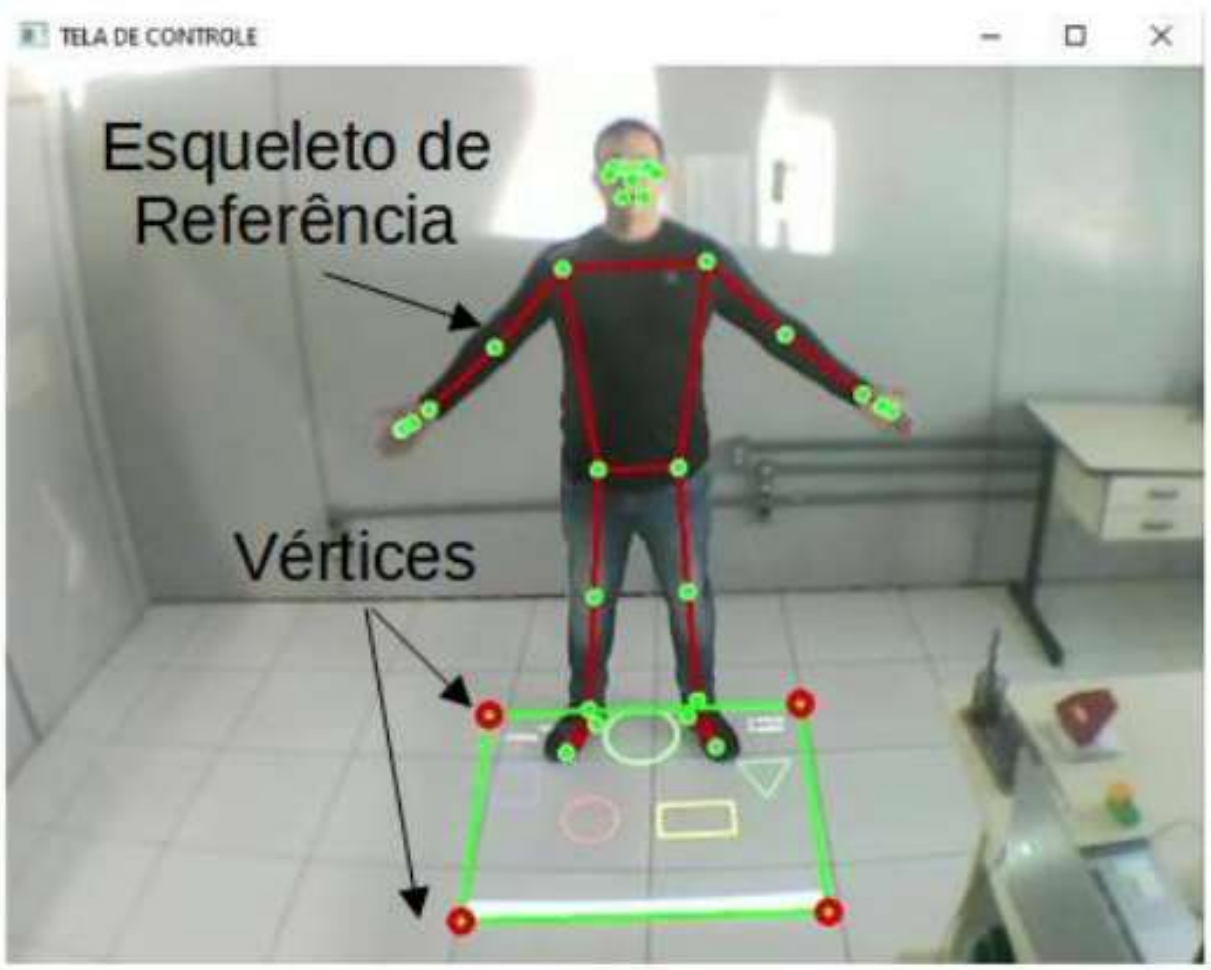
Agenda 13:30 às 17:30 (4 horas)

1. Pygame
2. **Console T-TEA**
3. OpenCV
4. MediaPipe
5. Pygame + OpenCV + MediaPipe

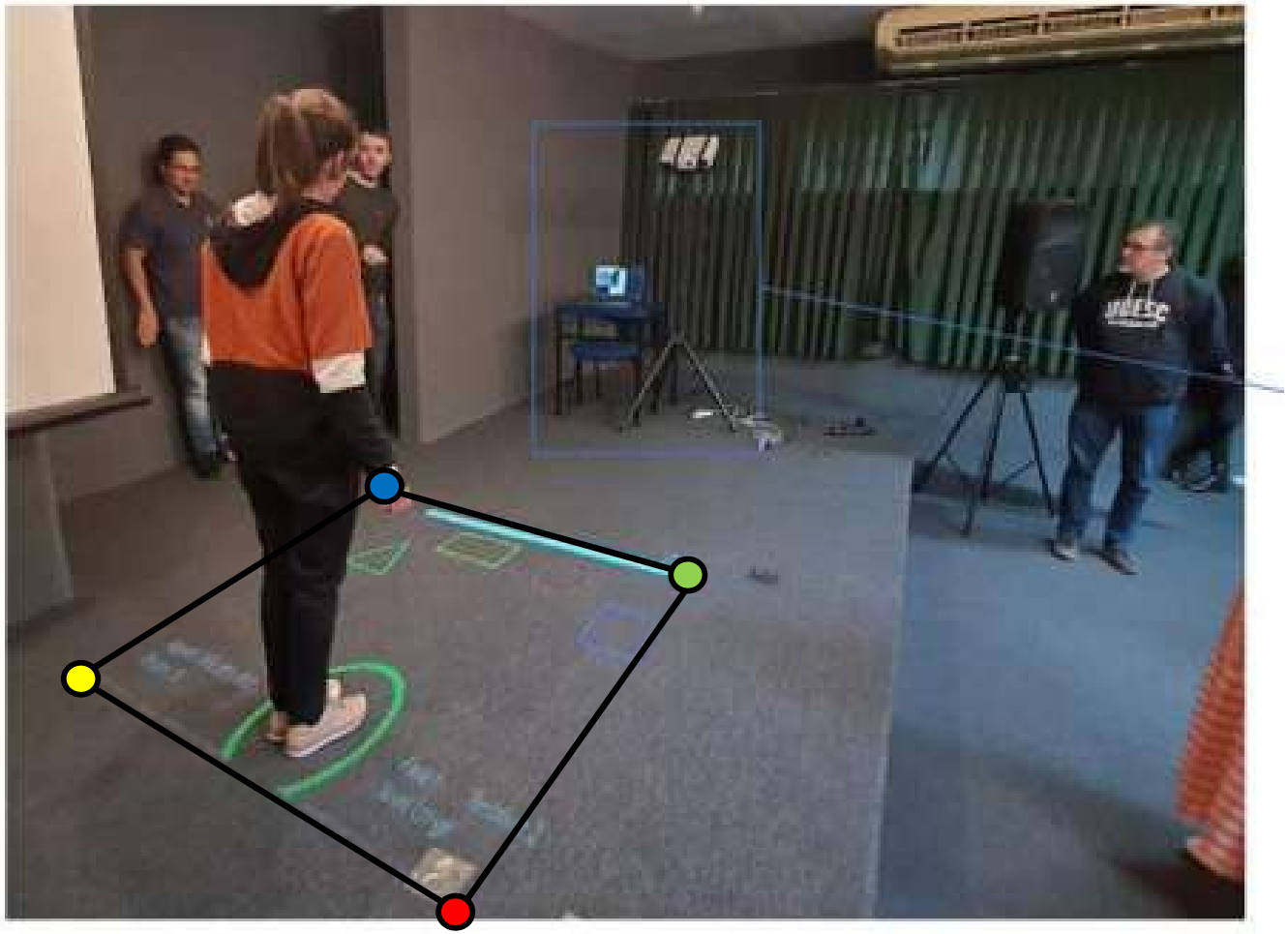
T-TEA

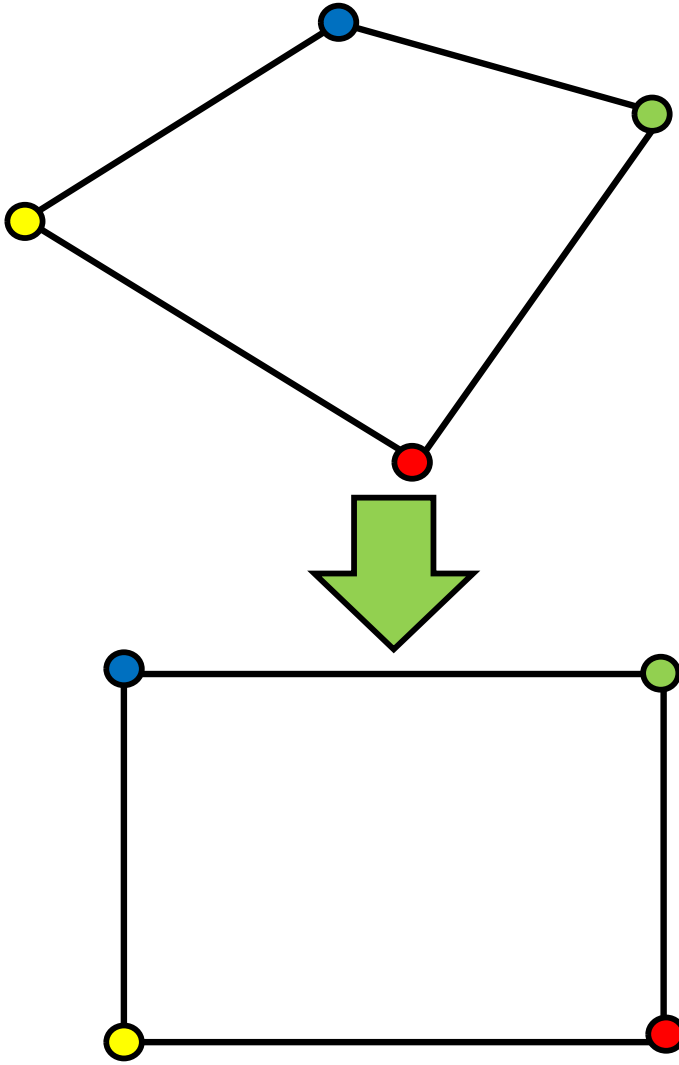


T-TEA – Visão Computacional

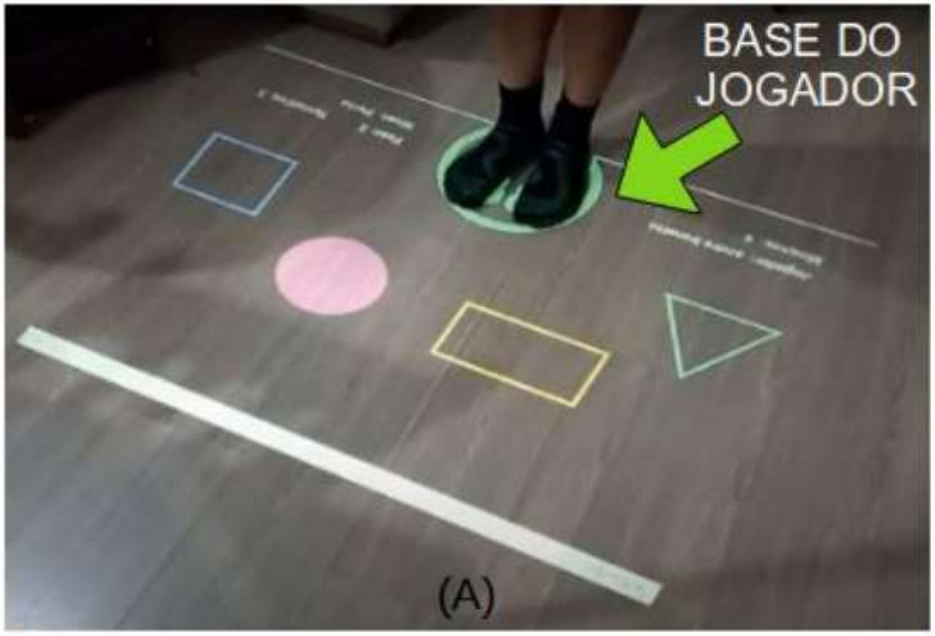


T-TEA – Perspectiva

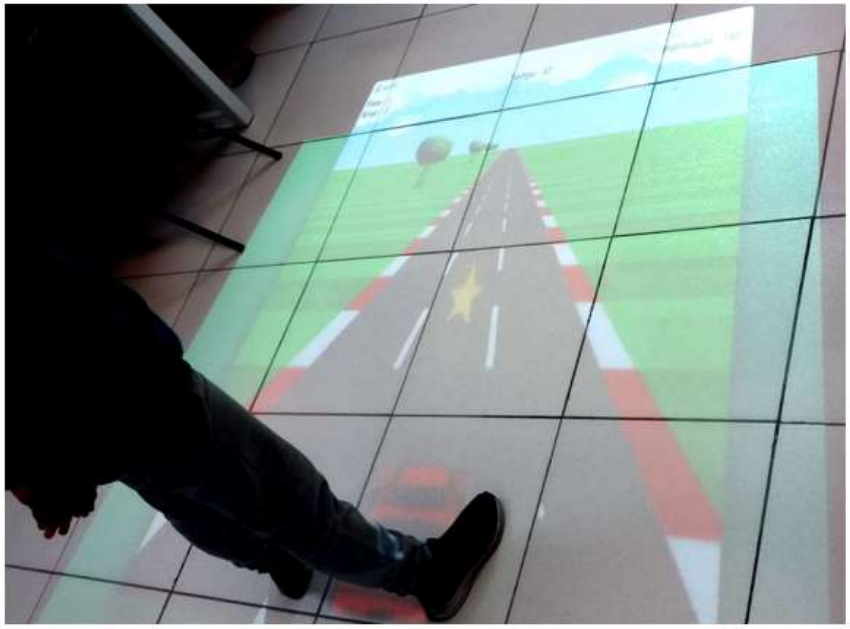




T-TEA Jogos

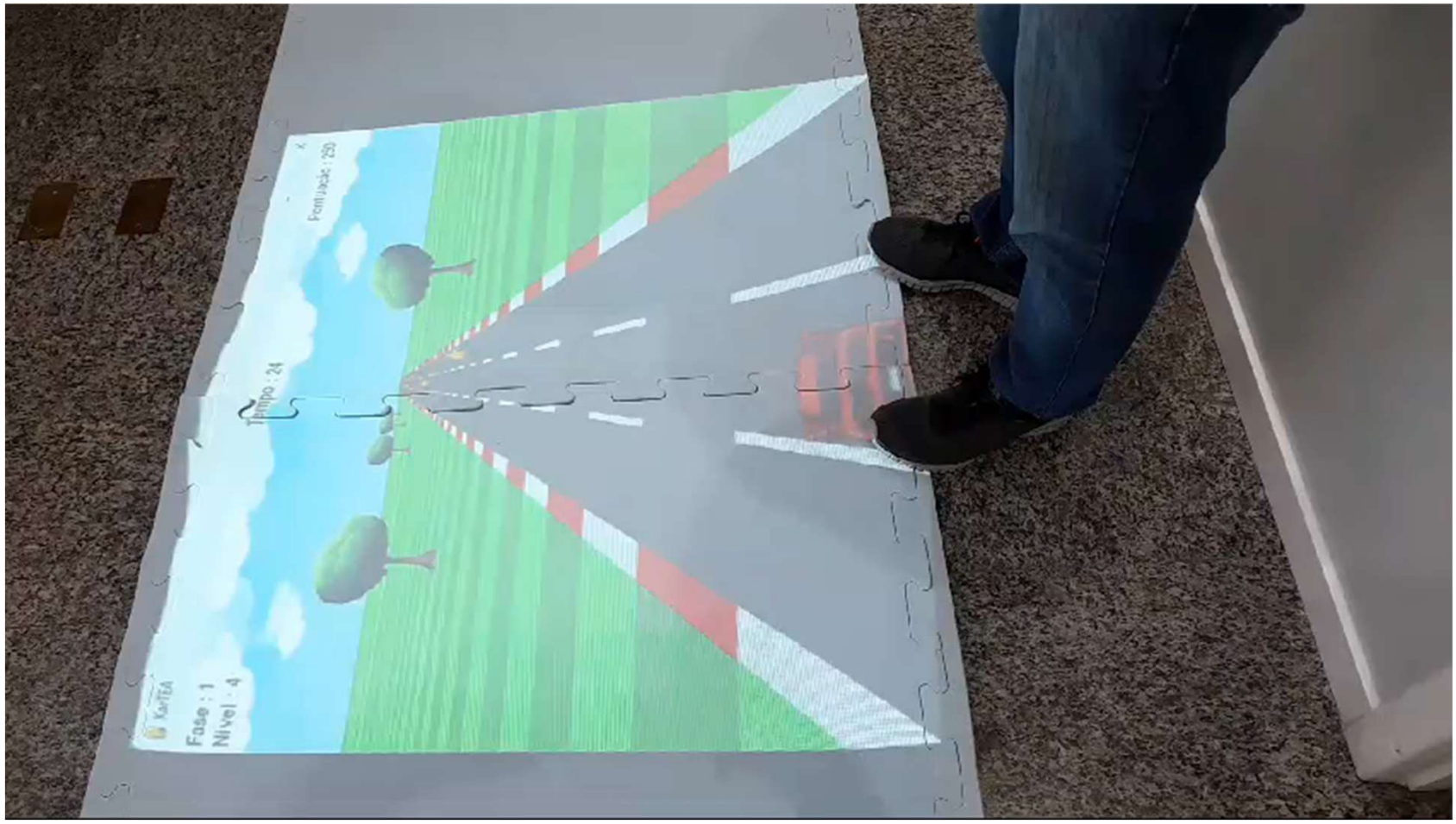


RepeTEA

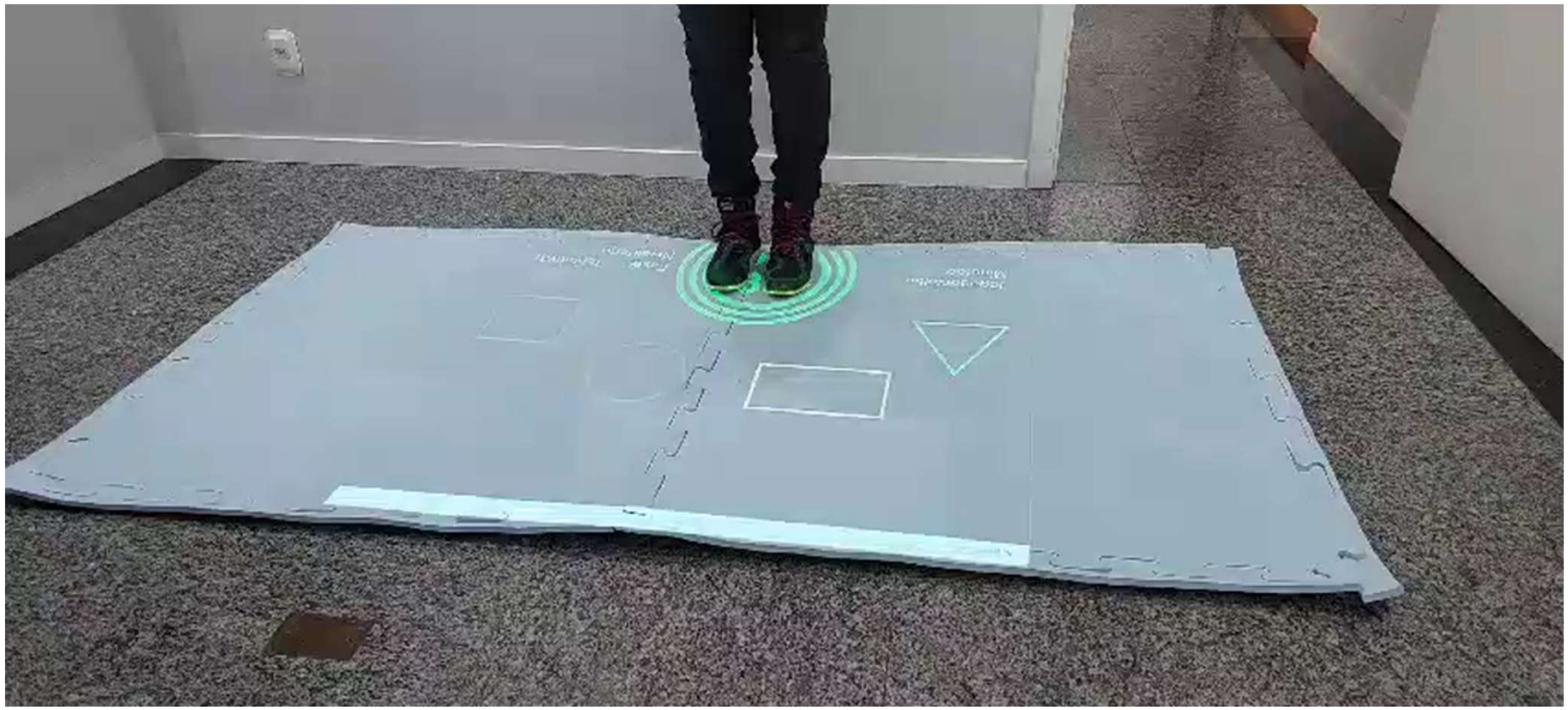


KarTEA

KarTEA - Cenário Dinâmico

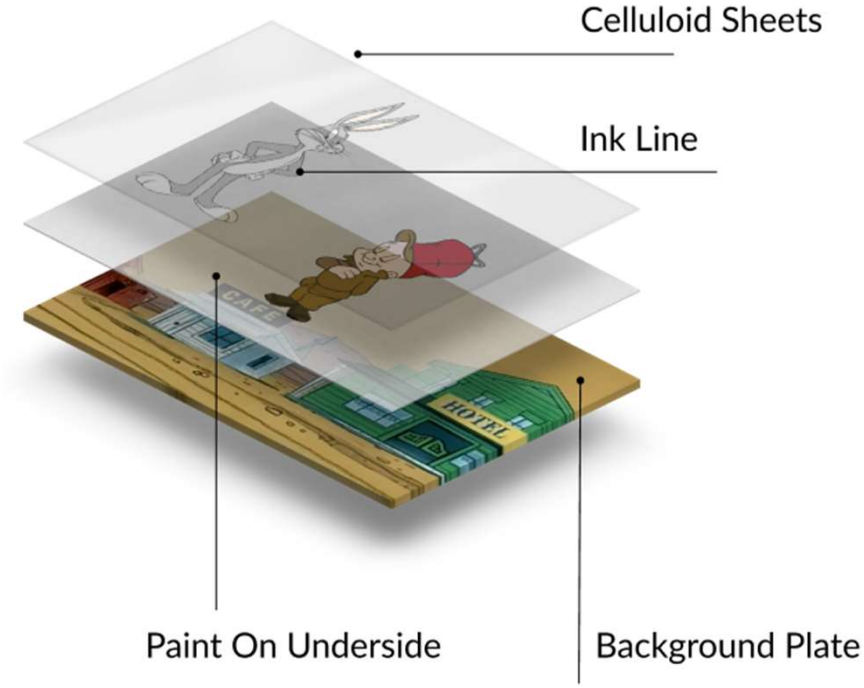


RepeTEA - Cenário Estático



Cenário Estático

Basicamente é um jogo por sobreposição de imagens estáticas, sem a necessidade de prover a sensação de movimento dos elementos do jogo.



Ícone*

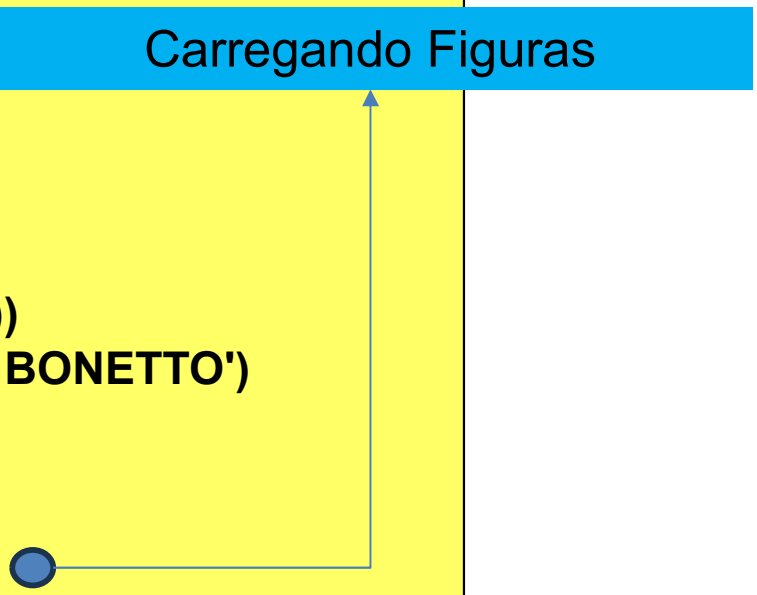
```
import pygame
import time

pygame.init()
largura_tela = 630
altura_tela = 480
tela= pygame.display.set_mode((largura_tela,altura_tela))
pygame.display.set_caption('MEU PYANO - POR ANDRÉ BONETTO')

icon=pygame.image.load('figura_3_verde.png')
pygame.display.set_icon(icon)
run = True

while run:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run=False
 pygame.quit()
```

Atualizando o Ícone

*Arquivo:
10_ícone.py

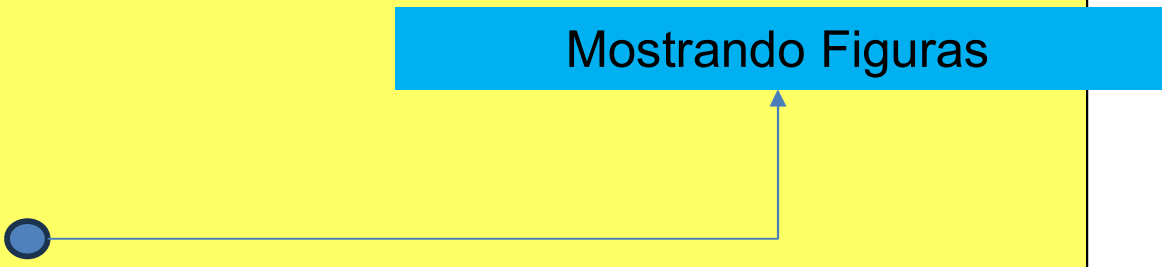
Carregando Imagens Estáticas

```
import pygame
import time

pygame.init()
largura_tela = 630
altura_tela = 480
tela= pygame.display.set_mode((largura_tela,altura_tela))
pygame.display.set_caption('MEU PYANO - POR ANDRÉ BONETTO')

figura_1=pygame.image.load('figura_1_vermelho.png')
figura_2=pygame.image.load('figura_2_amarelo.png')
figura_3=pygame.image.load('figura_3_verde.png')

run = True
```

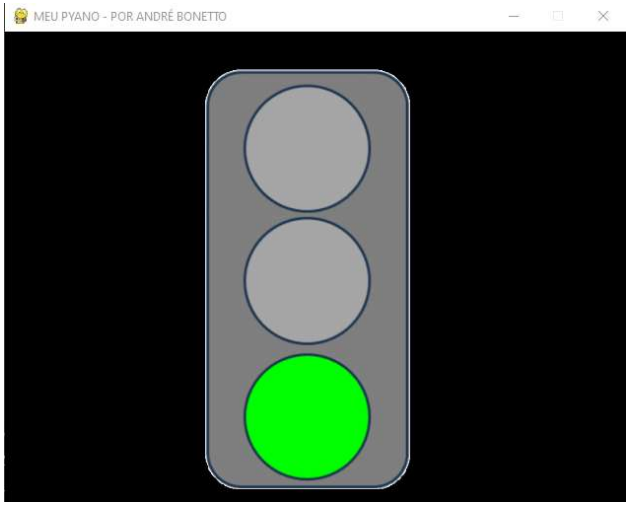


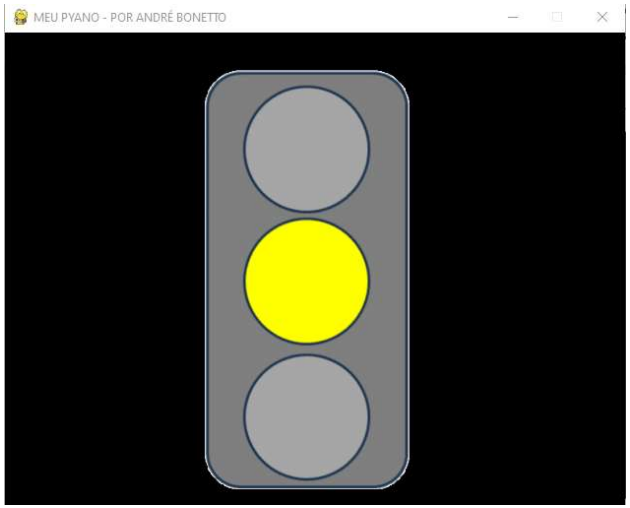
Carregando Imagens Estáticas

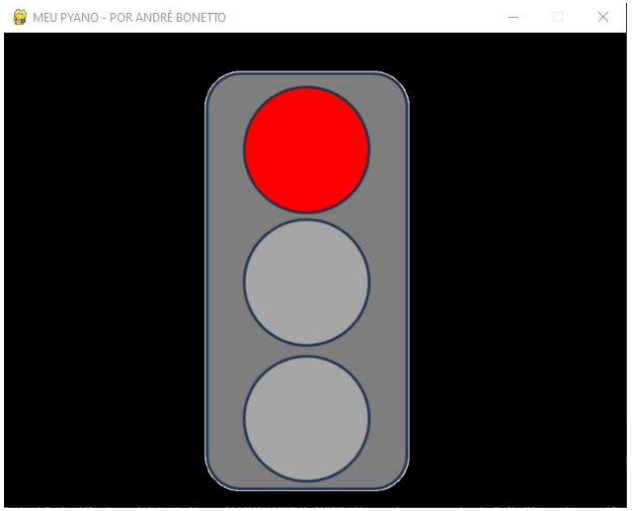
```
while run:

    tela.blit(figura_3, (0, 0))
    pygame.display.update()
    time. sleep(1)

    tela.blit(figura_2, (0, 0))
    pygame.display.update()
    time. sleep(1)

    tela.blit(figura_1, (0, 0))
    pygame.display.update()
    time. sleep(1)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run=False
pygame.quit()
```



Exercício*







*Arquivo:
10_sinaleiro.py

Agenda 13:30 às 17:30 (4 horas)

1. Pygame
2. Console T-TEA
3. **OpenCV**
4. MediaPipe
5. Pygame + OpenCV + MediaPipe

Bibliotecas

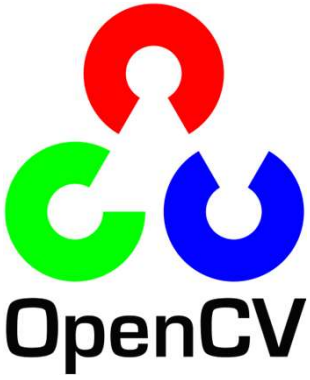
- **OpenCV** é a maior biblioteca de visão computacional do mundo. É de código aberto, contém mais de 2.500 algoritmos e é operado pela organização sem fins lucrativos Open Source Vision Foundation.

- **NumPy** é um projeto de código aberto que permite computação numérica com Python. Foi criado em 2005 com base nos primeiros trabalhos das bibliotecas Numeric e Numarray. NumPy é um software 100% de código aberto e gratuito para todos usarem.

Bibliotecas

- **OpenCV**

pip install opencv-python --user



Transformação de Perspectiva

Funções OpenCV

As funções nesta seção realizam diversas transformações geométricas de imagens 2D. Eles não alteram o conteúdo da imagem, mas deformam a grade de pixels e mapeiam essa grade deformada para a imagem de destino.

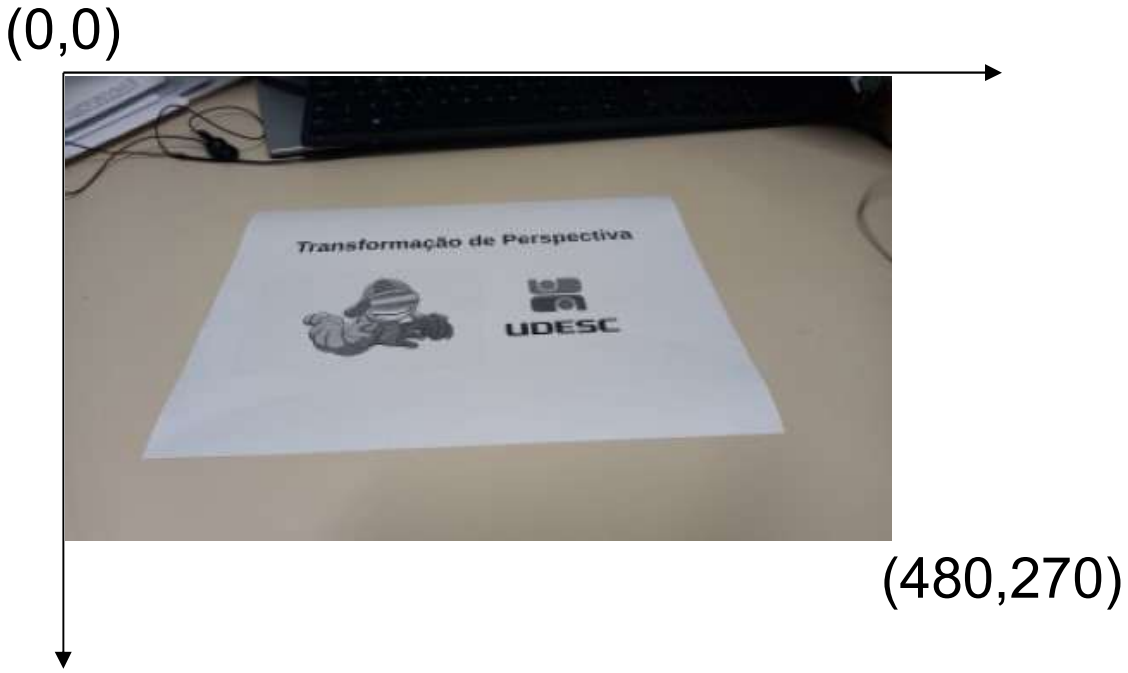
- **getPerspectiveTransform:** Calcula uma transformação de perspectiva a partir de quatro pares de pontos correspondentes.

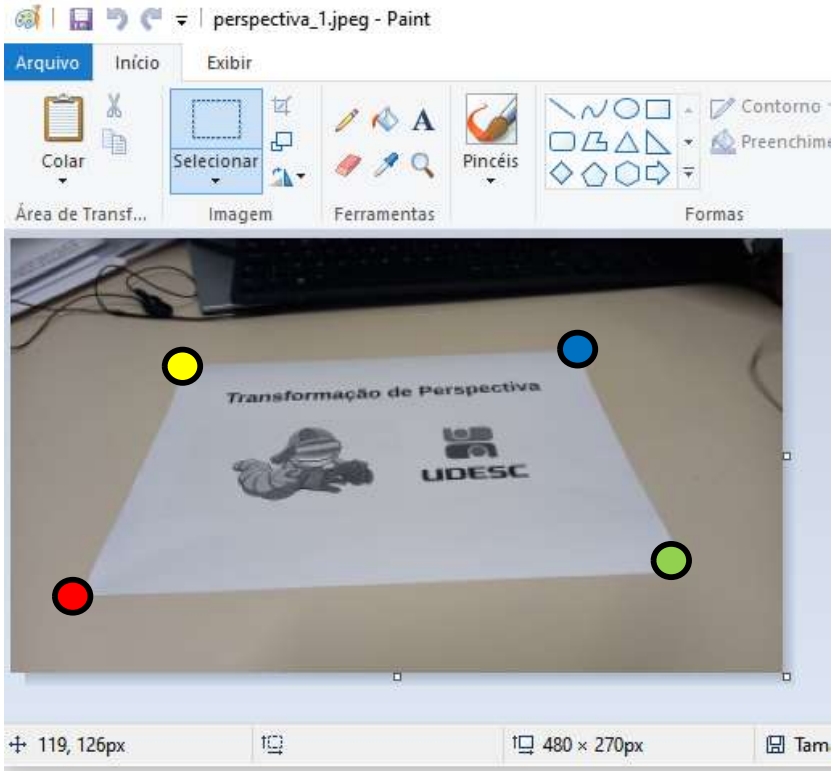
- **warpPerspective**: Aplica uma transformação de perspectiva a uma imagem.

Fonte: https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html

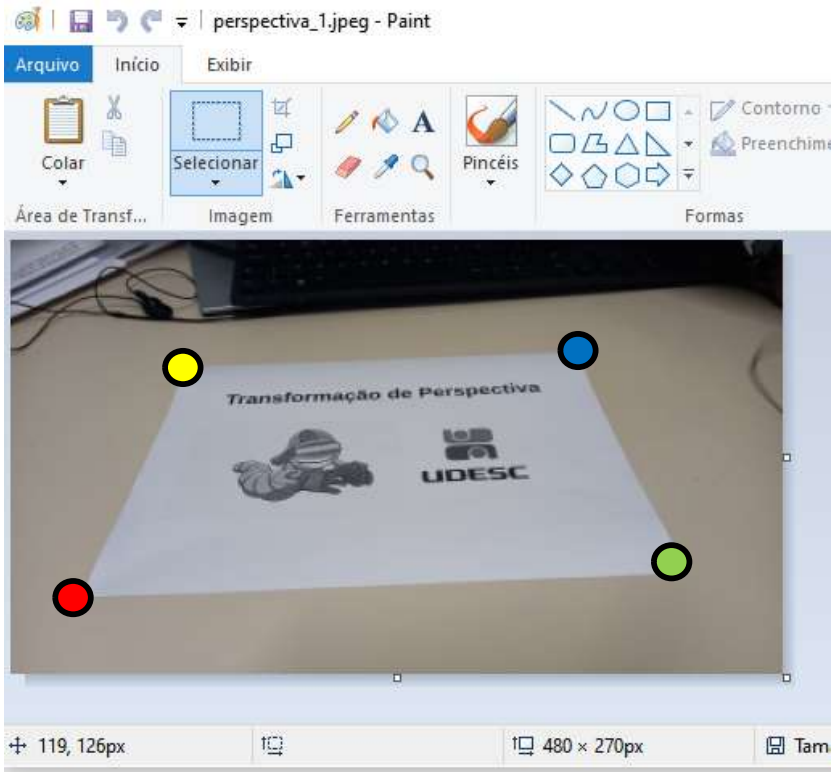
Transformação de Perspectiva “Manual”

Abra a figura no MS Paint e anote as coordenadas dos vértices





Transformação de Perspectiva “Manual”



Abra a figura no MS Paint e anote as coordenadas dos vértices:

- [110, 78]
- [352, 70]
- [44, 222]
- [420, 208]

Transformação de Perspectiva “Manual”

```
import cv2
import numpy as np

imagem = cv2.imread('perspectiva_1.jpeg')
largura, altura = 480, 270

pontos_1 = np.float32([[110, 78],[352, 70], [44, 222], [420, 208]])
pontos_2 = np.float32([[0,0], [largura,0],[0,altura],[largura,altura]])
matrix = cv2.getPerspectiveTransform(pontos_1,pontos_2)
imagem_corrigida = cv2.warpPerspective(imagem,matrix,(largura,altura))

cv2.circle(imagem,(int(pontos_1[0][0]),int(pontos_1[0][1])),5,(0,255,255),cv2.FILLED)
cv2.circle(imagem,(int(pontos_1[1][0]),int(pontos_1[1][1])),5,(255,0,0),cv2.FILLED)
cv2.circle(imagem,(int(pontos_1[2][0]),int(pontos_1[2][1])),5,(0,0,255),cv2.FILLED)
cv2.circle(imagem,(int(pontos_1[3][0]),int(pontos_1[3][1])),5,(0,255,0),cv2.FILLED)


cv2.imshow("Imagem Original", imagem)
cv2.imshow("Imagem Corrigida", imagem_corrigida)
cv2.waitKey(0)
```

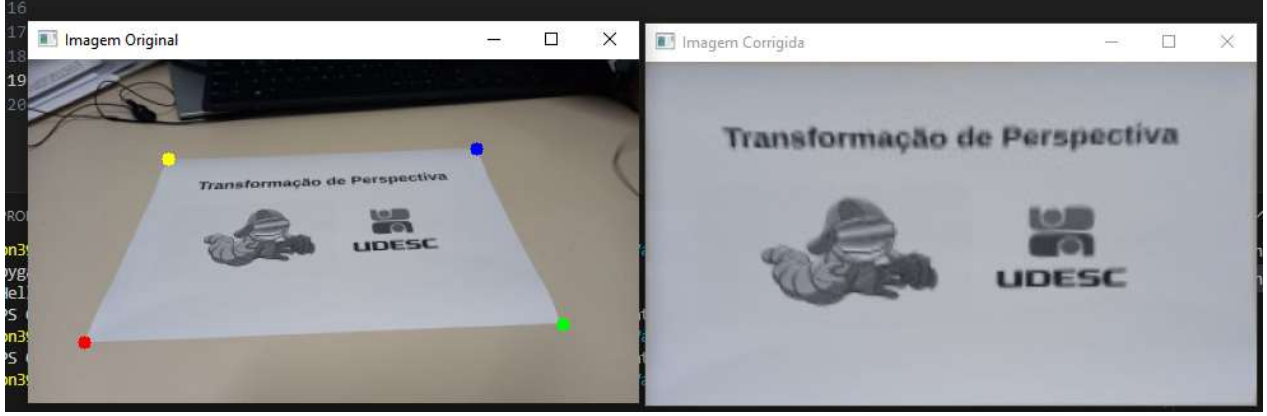
Imagem original e a tela final que será projetada a transformação

Transformação

Vértices

Transformação de Perspectiva Manual*

```python
import cv2
import numpy as np

imagem = cv2.imread('perspectiva_1.jpeg')
largura, altura = 480, 270

pontos_1 = np.float32([[110, 78],[352, 70], [44, 222], [420, 208]])
pontos_2 = np.float32([[0,0], [largura,0],[0,altura],[largura,altura]])
matrix = cv2.getPerspectiveTransform(pontos_1,pontos_2)
imagem_corrigida = cv2.warpPerspective(imagem,matrix,(largura,altura))

cv2.circle(imagem,(int(pontos_1[0][0]),int(pontos_1[0][1])),5,(0,255,255),cv2.FILLED)
cv2.circle(imagem,(int(pontos_1[1][0]),int(pontos_1[1][1])),5,(255,0,0),cv2.FILLED)
cv2.circle(imagem,(int(pontos_1[2][0]),int(pontos_1[2][1])),5,(0,0,255),cv2.FILLED)
cv2.circle(imagem,(int(pontos_1[3][0]),int(pontos_1[3][1])),5,(0,255,0),cv2.FILLED)
```

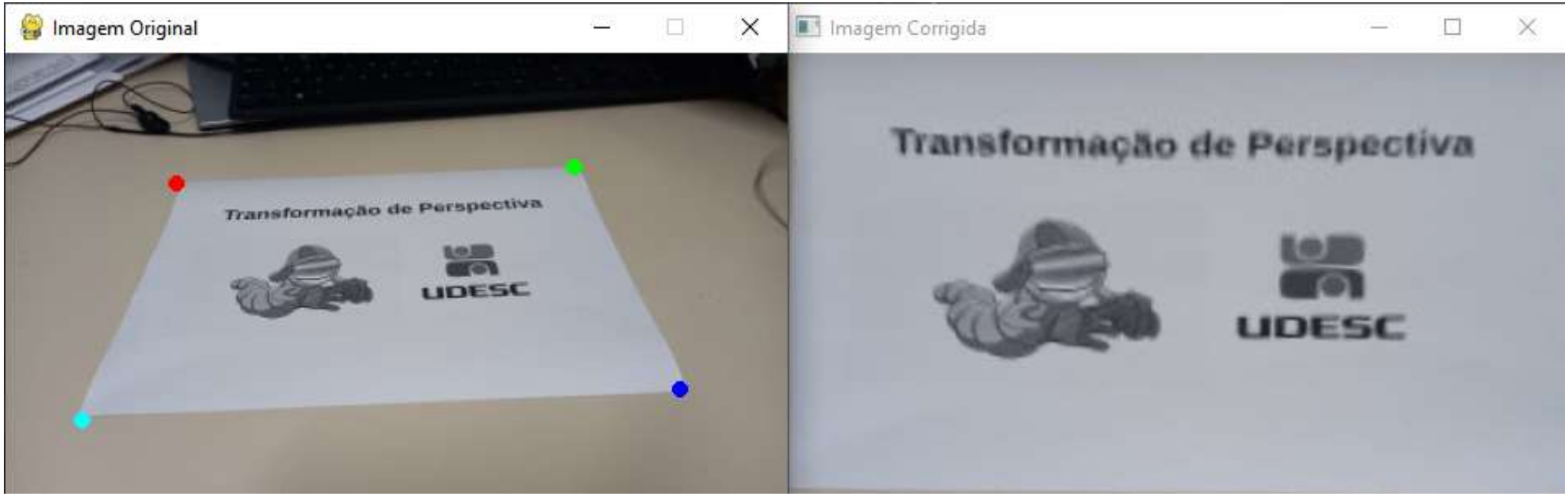


*Arquivo:
11_perspectiva_manual.py

Transformação de Perspectiva “Mouse”

Neste exemplo utilizaremos os eventos de mouse para determinar os vértices e vamos unir Pygame e OpenCV.

Vamos por partes para ficar mais fácil entender cada função do código:



Transformação de Perspectiva “Mouse”

```
import pygame
import cv2
import numpy as np

pygame.init()

largura = 480
altura = 270
tela= pygame.display.set_mode((largura,altura))
pygame.display.set_caption('Imagem Original')
vertices = np.zeros((4, 2), int)
contador = 0

imagem_pygame=pygame.image.load('perspectiva_1.jpeg')
imagem_original = cv2.imread('perspectiva_1.jpeg')
run = True

def background (imagem_pygame):
    tela.blit(imagem_pygame, (0, 0))
```

Matriz 4 linhas x 2 colunas onde serão armazenados os vértices

Transformação de Perspectiva “Mouse”

```
while run:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run=False

        if event.type == pygame.MOUSEBUTTONDOWN:
            if contador<4:
                pos = pygame.mouse.get_pos()
                vertices[contador] = pos
                contador=contador+1

    background (imagem_pygame)
```

Será admitido até 4 cliques na tela. Lembrando que a sequência deve corresponder o que foi colocado na matriz de transformação

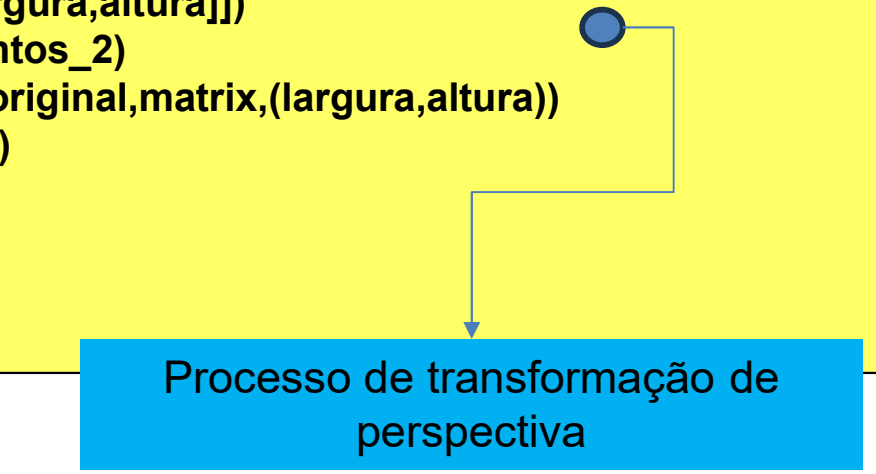
Transformação de Perspectiva “Mouse”

```
if contador>0:
    pygame.draw.circle(tela, (255,0,0), (vertices[0]), 5)
if contador>1:
    pygame.draw.circle(tela, (255,0,0), (vertices[0]), 5)
    pygame.draw.circle(tela, (0,255,0), (vertices[1]), 5)
if contador>2:
    pygame.draw.circle(tela, (255,0,0), (vertices[0]), 5)
    pygame.draw.circle(tela, (0,255,0), (vertices[1]), 5)
    pygame.draw.circle(tela, (0,255,255), (vertices[2]), 5)
if contador>3:
    pygame.draw.circle(tela, (255,0,0), (vertices[0]), 5)
    pygame.draw.circle(tela, (0,255,0), (vertices[1]), 5)
    pygame.draw.circle(tela, (0,255,255), (vertices[2]), 5)
    pygame.draw.circle(tela, (0,0,255), (vertices[3]), 5)
```

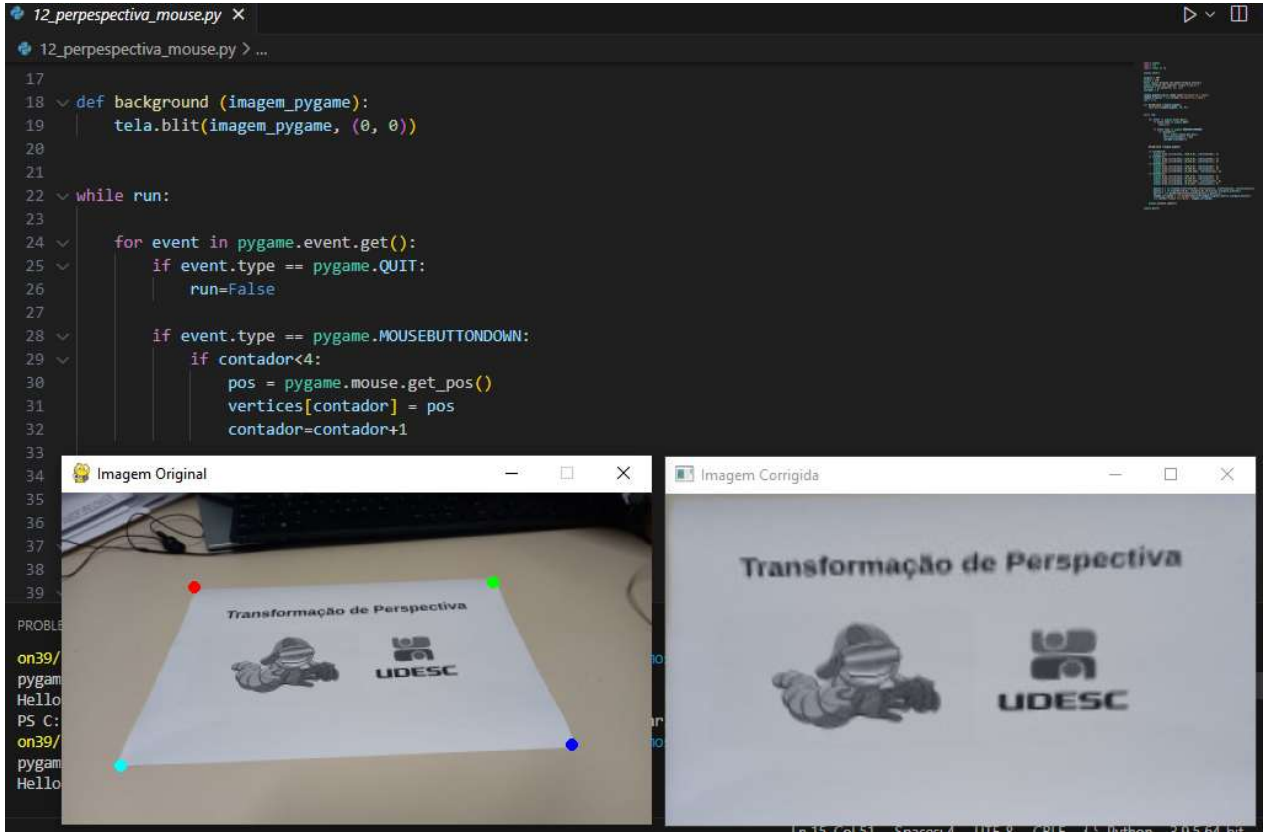
Apenas para desenhar os círculos nos vértices

Transformação de Perspectiva “Mouse”

```
    pontos_1 = np.float32([[vertices[0]],[vertices[1]], [vertices[2]], [vertices[3]]])
    pontos_2 = np.float32([[0,0], [largura,0],[0,altura],[largura,altura]])
    matrix = cv2.getPerspectiveTransform(pontos_1,pontos_2)
    imagem_corrigida = cv2.warpPerspective(imagem_original,matrix,(largura,altura))
    cv2.imshow("Imagem Corrigida", imagem_corrigida)

  pygame.display.update()

pygame.quit()
```



Transformação de Perspectiva Mouse*



*Arquivo:
12_perpespectiva_mouse.py

Agenda 13:30 às 17:30 (4 horas)

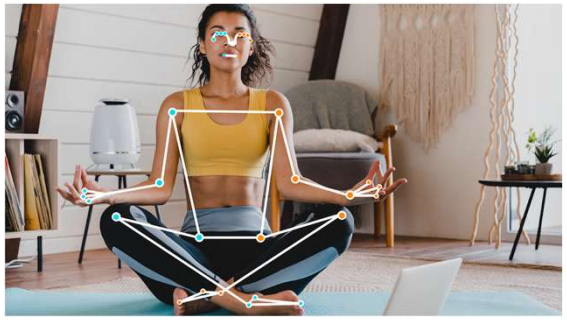
1. Pygame
2. Console T-TEA
3. OpenCV
4. **MediaPipe**
5. Pygame + OpenCV + MediaPipe

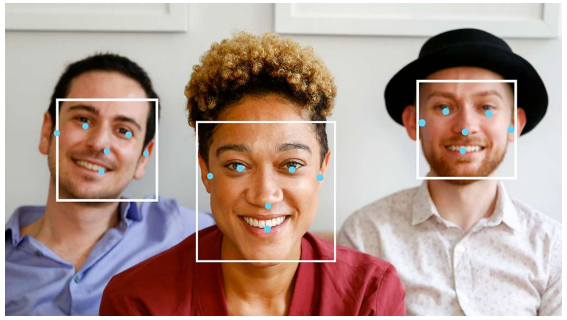
MediaPipe

O MediaPipe fornece um conjunto de bibliotecas e ferramentas para você aplicar técnicas de inteligência artificial (IA) e machine learning (ML).

https://ai.google.dev/edge/mediapipe/solutions/guide?hl=pt-br

MediaPipe

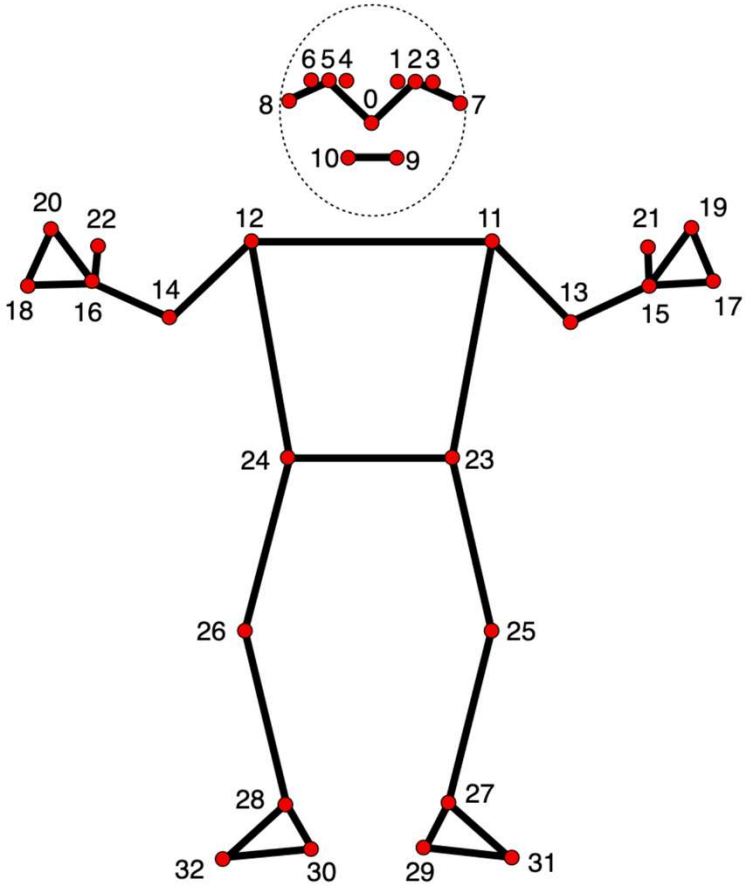




Solução	Android	Web	Python	iOS	Personalizar modelo
API LLM Inference	●	●		●	●
Detecção de objetos	●	●	●	●	●
Classificação de imagens	●	●	●	●	●
Segmentação de imagens	●	●	●		
Segmentação interativa	●	●	●		
Detecção de pontos de referência manualmente	●	●	●	●	
Reconhecimento de gestos	●	●	●	●	●
Incorporação de imagens	●	●	●		
Detecção facial	●	●	●	●	
Detecção de pontos de referência do rosto	●	●	●		
Estilização de rostos	●	●	●		●
Detecção de pontos de referência	●	●	●		
Geração de imagens	●				●
Classificação de texto	●	●	●	●	●
Embedding de texto	●	●	●		
Detector de idioma	●	●	●		
Classificação de áudio	●	●	●		

MediaPipe - Pose

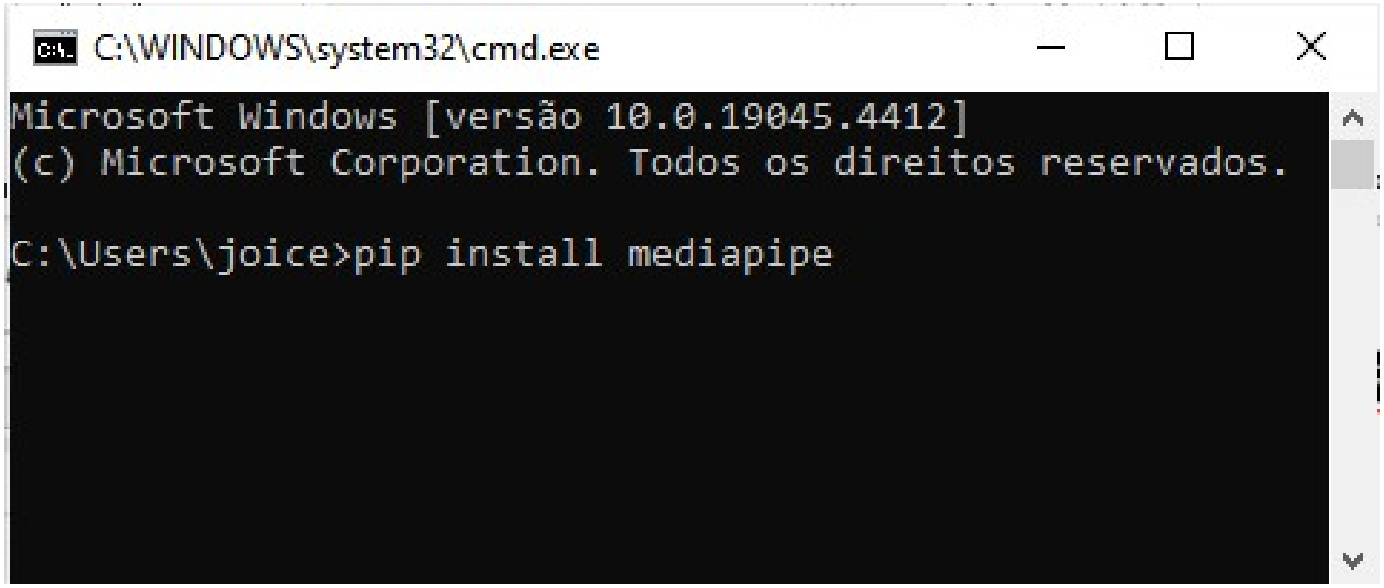
0 - nose
1 - left eye (inner)
2 - left eye
3 - left eye (outer)
4 - right eye (inner)
5 - right eye
6 - right eye (outer)
7 - left ear
8 - right ear
9 - mouth (left)
10 - mouth (right)
11 - left shoulder
12 - right shoulder
13 - left elbow
14 - right elbow
15 - left wrist
16 - right wrist



17 - left pinky
18 - right pinky
19 - left index
20 - right index
21 - left thumb
22 - right thumb
23 - left hip
24 - right hip
25 - left knee
26 - right knee
27 - left ankle
28 - right ankle
29 - left heel
30 - right heel
31 - left foot index
32 - right foot index

MediaPipe - Instalando

pip install mediapipe --user



MediaPipe - Configurando

Importando a biblioteca MediaPipe

```
import cv2
import mediapipe as mp
import numpy as np

mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose

cap = cv2.VideoCapture('video_para_testar_mediapipe.mp4')
```

Soluções para o módulo de estimativa de poses

Fonte de imagem, pode ser a câmera, uma imagem ou um vídeo

MediaPipe – Estimando Poses

Definições de confiança para o modelo

```
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:

    while cap.isOpened():
        ret, frame = cap.read()
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False
        results = pose.process(image)
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS)


        cv2.imshow('MediaPipe', image)
        cv2.imshow('Original', frame)
```

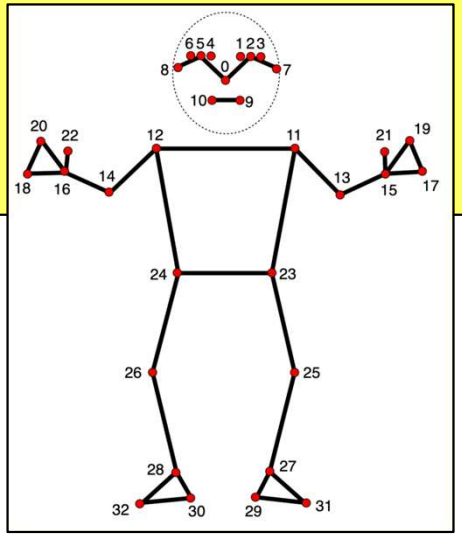
Processa a imagem

Desenha as juntas e conexões

MediaPipe – Extraindo coordenadas

Resultado das coordenadas normalizadas para um dos 33 pontos

```
try:
        landmarks = results.pose_landmarks.landmark
        x_pose = landmarks[mp_pose.PoseLandmark.NOSE.value].x
        y_pose = landmarks[mp_pose.PoseLandmark.NOSE.value].y
        print(x_pose,y_pose)

except:
        pass
```



MediaPipe – Exemplo*

*Arquivo:
13_mediapipe.py

Agenda 13:30 às 17:30 (4 horas)

1. Pygame
2. Console T-TEA
3. OpenCV
4. MediaPipe
5. **Pygame + OpenCV + MediaPip**

Pygame + OpenCV + MediaPipe*

Substitua a posição dos mouse pelos resultados do processamento do MediaPipe

pos= ((x_pose*largura_tela),(y_pose*altura_tela))

Copie e cole o código do PYANO_GEOMETRIAS

Pygame + OpenCV + MediaPipe

*Arquivo:
PYANO_MEDIAPIPE.py

Para Exergames

Alexandre Melo
André Bonetto

24/09/2025
17:00 às 21:00h
Laboratório F101

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA